Tracking through Containers and Occluders in the Wild

Supplementary Material

A. Dataset Details

For our training set Kubric Random, all scenes are generated based on the *MOVi-F*[1] template code [8], but with several modifications. Backgrounds are chosen randomly from the Polyhaven HDRI collection [11], and all objects originate from Google Scanned Objects (GSO) [7]. Every scene spawns s static objects lying on the ground, and d dynamic objects falling down when the video starts. s is uniformly randomly chosen between 4 and 24 (inclusive), while d is uniformly randomly chosen between 2 and 12 (inclusive).

In order to increase the frequency of containment, we manually scan the GSO library to designate 114 out of 1,032 GSO assets as *containers*, which can be either deep or shallow. For each scene, at least three out of the s static objects must be containers, and while object sizes are chosen randomly, we also make containers slightly bigger on average. An assortment of examples is shown in Figure 1.

The most time-consuming part of the simulation is generating the X-ray segmentation mask $\boldsymbol{m}_a \in [0,1]^{T\times H\times W\times K}$, which is used for supervision as it exposes all pixels of all K instances separately over time, regardless of occlusion. This is done by running the PyBullet physics simulation [5] once, thus letting the object interactions develop over time within the dynamic scene, then rendering the input video via Blender [3] with all instances present, following [8]. Next, we isolate each object by turning off the visibility of all other objects (they are essentially temporarily removed from existence), and rendering those videos again separately to iteratively produce one channel of $\boldsymbol{m}_a$ at a time.

Finally, even though the frame rate of video clips in the Rubric benchmark is variable (*i.e.* between 4 and 30), rendering of all Kubric simulations happens at a single fixed value of 12 FPS.

To construct the Kubric Random dataset, consisting of 4,000 videos of 36 frames each with spatial dimension 480×360 along with RGB information, depth maps, and segmentation maps ($\boldsymbol{m}_v$ and $\boldsymbol{m}_a$), 256 AMD EPYC 7763 CPU cores worked for 30 days.

A.1. Mass Estimation

Mass plays an important role in determining the outcome of object dynamics and interactions. While GSO provides a diverse collection of high-quality scanned 3D models for household items, physical properties such as mass and friction were not captured for many objects [7]. In Kubric

[1]This is the same as *MOVi-E*, but with a small degree of motion blur added to the video recorded by the virtual camera.

Figure 1. **Containers in GSO.** We mark roughly 11% of the assets in Google Scanned Objects [7] to be containers, which are spawned more often than average compared to other object types in Kubric Random.

MOVi-F, a constant density assumption is therefore made by default to estimate mass from volume [8]. In an attempt to increase the realism of our training data, we leverage GPT-3 [4] to produce rough estimates of the mass of every object in the GSO library based on its description and metadata. This is illustrated in Figure 2. In practice, we calculate and apply the geometric mean of the original and LLM-estimated mass, because the numbers provided by GPT-3 are, qualitative speaking, not always very accurate.

B. Network Implementation Details

B.1. AOT

Since AOT is designed for VOS, we keep the entire pipeline of the AOT model intact for fairness. Following [10], at training time, a context window of 5 frames is fed into the model for a single training step, while at test time, the target object mask is propagated throughout the entire video clip from start to end.

B.2. TCOW

TCOW is a modification of the TimeSFormer network, which operates by processing a number of chunks of space-time patches into a transformer [1, 9]. Specifically, we concatenate the input video and the query mask along the channel axis to form $(\boldsymbol{x}, \boldsymbol{m}_q) \in \mathcal{R}^{T\times H\times W\times 4}$ (here, $\boldsymbol{m}_{q,t} = 0$ for all $t \geq 1$ as only the first frame is labeled). Similarly to Vision Transformer [6], the resulting set of frames is decomposed into $N = T \times h \times w$ small image patches of size $16 \times 16 \times 4$ each, with $h = \frac{H}{16}$, $w = \frac{W}{16}$. After a per-patch linear projection, an input sequence of N embeddings of dimensionality 768 is fed into a transformer, where we subse-

Category: *Consumer Goods*
Description: *Epson UltraChrome T0548 Ink Cartridge, Matte Black - 1-pack*
Height: *11.54 cm.*
[Mass / Weight]: *[0.065 kg. / 0.065 kg. / 0.09 kg. / 0.14 kg.] + random text*

Category: *None*
Description: *Vtech Roll & Learn Turtle*
Height: *11.75 cm.*
[Mass / Weight]: *[0.1 kg. / 0.09 kg. / 0.1 kg. / 0.5 kg.] + random text*

Figure 2. **Estimating mass for objects used in Kubric simulations.** We perform text completion with a large language model. Specifically, we query OpenAI GPT-3 (text-davinci-002) [4] twice for mass, twice for weight, and average the four numerical outputs after appropriate unit conversions. The image is shown for visualization only, and is not fed to the language model. The underlined text represents the four actual completion outputs made by GPT-3. The italic parts of the input are derived from the available metadata of each asset, and this procedure is repeated for all 1,032 GSO objects.

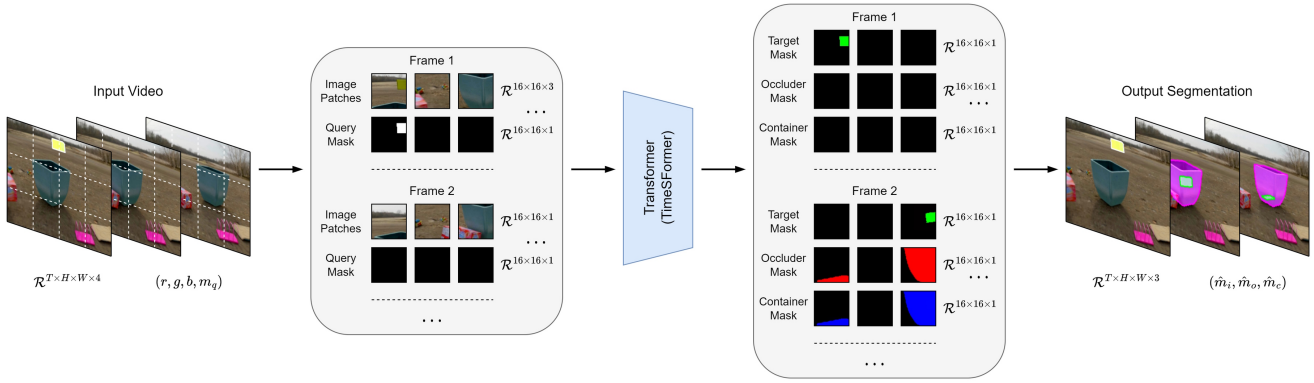

Figure 3. **TCOW architecture.** We apply the standard TimeSFormer backbone onto the input video $(\boldsymbol{x}, \boldsymbol{m}_q)$ following a spacetime divided attention scheme [1], but interpret the tokens after the transformer as patches for the predicted output masks. (Multiple channels belonging to the same patch are shown in separate tiles for clarity.)

quently apply repeated multi-head self-attention blocks on these tokens.

The output sequence is treated as a spatiotemporal feature map for the purpose of dense video segmentation. Each element after the last attention layer is linearly projected back to image space, resulting in a set of patches of $16 \times 16 \times 3$, where the last dimension represents the predicted triplet of masks $(\hat{\boldsymbol{m}}_t, \hat{\boldsymbol{m}}_o, \hat{\boldsymbol{m}}_c)$. The vectors are composed in the same order as they were decomposed at the input side. The classification token is ignored and there is no pooling. A diagram is shown in Figure 3.

B.3. Learning and Supervision

We train the TCOW model for tracking objects through occlusion and containment by producing segmentation masks for each type. The network f (as defined in Equation 1 in the main text) accepts a single query instance at a time, which makes a binary cross-entropy objective $\mathcal{L}_{BCE}$ between every output channel $\hat{\boldsymbol{m}}$ and its corresponding ground truth $\boldsymbol{m}$ a logical starting point.

Since the number of frames where the target is occluded is typically smaller than the number of frames where the target is visible in our training set, we scale $\mathcal{L}_{BCE}$ by a factor $1+(\beta-1)o$, where $o \in [0, 1]$ is the occlusion fraction.

However, inspired by [10], we also combine $\mathcal{L}_{BCE}$ with two additional loss terms: (1) a bootstrapped variant $\mathcal{L}_{BCE,k}$ that focuses on a certain top fraction k of pixels in each example that incur the highest individual contributions to the loss $\mathcal{L}_{BCE}$, and (2) a soft Jaccard loss $\mathcal{L}_{\mathcal{J}}$ [2]. The terms are linearly combined and weighted as follows:

$$\mathcal{L}_{\boldsymbol{m}} = (\lambda_1 \mathcal{L}_{BCE} + \lambda_2 \mathcal{L}_{BCE,k} + \lambda_3 \mathcal{L}_{\mathcal{J}})(\hat{\boldsymbol{m}}, \boldsymbol{m}) \quad (1)$$

Finally, the total objective is a weighted sum over the three different output types predicted by f:

$$\mathcal{L} = \lambda_t \mathcal{L}_{\boldsymbol{m}_t} + \lambda_o \mathcal{L}_{\boldsymbol{m}_o} + \lambda_c \mathcal{L}_{\boldsymbol{m}_c} \quad (2)$$

where $\mathcal{L}_{\boldsymbol{m}_t}$ addresses the target instance mask, $\mathcal{L}_{\boldsymbol{m}_o}$ is for the main occluder mask, and $\mathcal{L}_{\boldsymbol{m}_c}$ is for the main container mask. The ground truth masks for the latter two ($\boldsymbol{m}_o$ and $\boldsymbol{m}_c$) are defined to be all-zero whenever there exists no occluder or container respectively, although for class balanc-

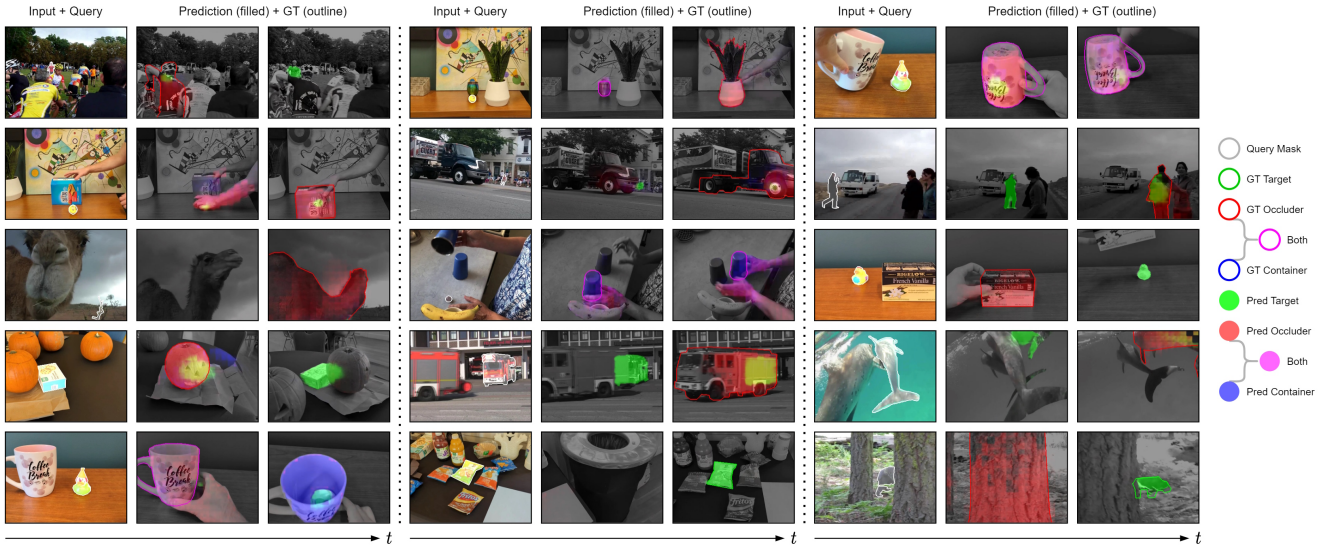

Figure 4. **Success cases for TCOW on Rubric.** All visualized predictions are made by the non-ablated TCOW network. This model performs particularly well on relatively simple cases of (total) occlusion and/or containment in the real world, despite being trained on synthetic data only. Some video clips with containers moving to a limited degree are also handled correctly (see middle center, or top right). However, more advanced examples of object permanence often result in failures, shown in Figure 5, demonstrating that a lot of room for improvement remains.

ing purposes, the loss is also weighted with a factor $\alpha < 1$ for those frames.

Augmentations during training consist of random color jittering (hue, saturation, brightness), random grayscale, random video reversal, random palindromes (*i.e.* playing clips forward and then backward, or vice versa), random horizontal flipping, and random cropping. We do not apply any augmentations at test time.

In Kubric Random, there are many possible objects with available annotations to track. At training time, we assign a difficulty score to every instance (that is visible in the first frame) based on its average occlusion fraction and how much motion it experiences over time. The query is then sampled randomly but non-uniformly, with preference given to the harder to track target objects. At test time, we measure and average metrics over the top four instances with the highest difficulty score per video. Other datasets (*i.e.* Kubric Containers plus all of Rubric) only have one designated target object per video clip.

In our experiments, we set $(T, H, W) = (30, 240, 320)$, $\beta = 5$, $(\lambda_1, \lambda_2, \lambda_3) = (0.2, 0.4, 0.4)$, $(\lambda_t, \lambda_o, \lambda_c) = (1.0, 0.5, 0.5)$, and $\alpha = 0.02$. The bootstrap fraction k is a function of time, and decreases linearly from 1 to 0.15 during the first 10% of training. We use the AdamW optimizer and train for 70 epochs, which takes 3 days on 2 NVIDIA RTX A6000 GPUs. Inference (without gradients) happens in 0.27 seconds for a single clip, which corresponds to roughly 110 FPS.

C. More Qualitative Results

Please see Figures 4, 5, and 6, as well as tcow.cs.columbia.edu for videos along with explanations. We recommend viewing the project webpage in a modern browser.

References

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021. 1, 2

[2] Jeroen Bertels, Tom Eelbode, Maxim Berman, Dirk Vandermeulen, Frederik Maes, Raf Bisschops, and Matthew B Blaschko. Optimizing the dice score and jaccard index for medical image segmentation: Theory and practice. In *International conference on medical image computing and computer-assisted intervention*, pages 92–100. Springer, 2019. 2

[3] Blender Online Community. Blender - a 3d modelling and rendering package, 2021. 1

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1, 2

[5] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 1

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner,

Figure 5. **Failure cases for TCOW on Rubric.** All visualized predictions are made by the non-ablated TCOW network. Multiple trends could be discerned among real-world scenarios where the model fails, which can roughly be summarized as: (1) identical containers, one of which is holding the target object, being shuffled around; (2) nested containment, *e.g.* when a mug is placed inside a larger box; (3) the occluder and occludee are visually very similar, *e.g.* people occluding people or animals occluding animals. By releasing this challenging benchmark to the community, we hope future work will be able to address these cases more successfully.

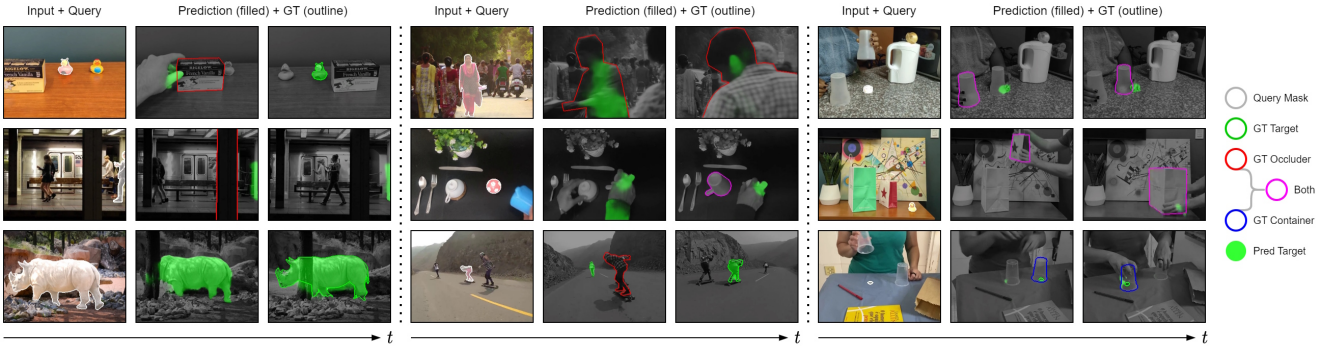

Figure 6. **Qualitative results for AOT on Rubric.** All visualized predictions are made by the non-ablated AOT network, and mirror Figure 6 in the main text. Although all models are trained on Kubric data with X-ray supervision, AOT often loses track as soon as total occlusion happens, and tends to jump to different instances or moving parts of the video (such as hands).

Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[7] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. *arXiv preprint arXiv:2204.11918*, 2022. 1

[8] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022. 1

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[10] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. *Advances in Neural Information Processing Systems*, 34:2491–2502, 2021. 1, 2

[11] Greg Zaal, Rob Tuytel, Rico Cilliers, James Ray Cock, Andreas Mischok, Sergej Majboroda, Dimitrios Savva, and Jurita Burger. Polyhaven: a curated public asset library for visual effects artists and game designers. https://polyhaven.com/hdris, 2021. 1